**Campos básicos**

**El fichero models.py**

Dentro de este fichero podemos definir nuestros objetos de negocio. También es posible tener un fichero por cada objeto de negocio que necesitemos. Debemos tener precaución a la hora de crearlo para saber como acceder a él posteriormente (con el namespace correcto)

El ORM de Odoo utiliza acceso jerárquico a los objetos, precediendo el nombre del módulo al objeto. Así por ejemplo tendremos:

* purchase.order: orden de compra
* account.invoice: factura
* sale.order: orden de venta

**Los objetos**

Todos los objetos que definamos extienden models.Model.

from odoo import models, fields  
  
class AModel(models.Model):  
  
    \_name = 'a.model.name'  
  
    field1 = fields.Char()

…

**\_name (obligatorio)**

Nombre del objeto de negocio en notación punto (es decir, con namespaces). Es decir, la tabla que se va a crear

**\_inherit**

Utilizado para heredar de un modelo. Lo veremos en detalle con posterioridad al estudiar los distintos tipos de herencia

**fields**

Propiedades del objeto de negocio.

Por defecto, la etiqueta visible un campo será su nombre con letra capital pero se puede especificar otra con el parámetro string.

field2 = fields.Integer(string=’an other field’)

El valor por defecto de un campo se pasan como parámetro en el fields

a\_field = fields.Char(default=’a value’)

Un field también puede llamar a una función para calcular su valor.

a\_field = fields.Char(default=compute\_default\_value)  
  
def compute\_default\_value(self):  
  
 return self.get\_value()

Se pueden consultar todas las propiedades de un objeto en la sección Model Reference.

**Los campos. Fields.**

Podemos utilizar los siguientes parámetros al definirlos:

* **string**: la etiqueta de campo que verá el usuario. Por defecto, el nombre del campo tipo Título.
* **help**: popup que se mostrará al usuario
* **readonly**: indica si es un campo de solo lectura. Por defecto False.
* **required**: indica si es un campo obligatorio. Por defecto False.
* **default**: indica el valor por defecto que debe tener el campo.

Respecto al tipo de datos que puede tener un Field puede ser **Char, Boolean, Integer, Float, Text, Selection, Html, Date, Datetime.** También dispone de campos relacionales que veremos en el siguiente apartado. Todos los tipos posibles para fields y sus parámetros en [Fields Reference](https://www.odoo.com/documentation/16.0/es/developer/reference/backend/orm.html#fields)

Por defecto, Odoo creará unos campos automáticamente en cada modelo: id (field), log\_access, create\_date (datetime), create\_uid(res.users), write\_date (datetime), write\_uid (res.users)

Un ejemplo de definición de un modelo sería:

class discografica(models.Model):  
    \_name = 'discografica.disco'  
    \_description = 'discografica.disco'  
  
    name = fields.Char(string='Título del disco',help='Título',required=True)  
    year = fields.Char(string='Año',help='Año',size=4)  
    genre = fields.Char(string='Género',help='Género')  
    artist = fields.Char(string='Grupo/Banda',help='Grupo')  
    cover = fields.Binary()    # Versión 14 y anteriores de Odoo  
 cover = fields.Image(max\_width=200, max\_height=200)  # Versiones posteriores a la 14

Tarea

En esta tarea vamos a continuar con nuestro *módulo soporte*. Vamos a comenzar a crear algunos modelos con unos pocos campos básicos.

En este *módulo de soporte*, se podrá solicitar una cita para uno de los servicios disponibles. Por tanto, vamos a necesitar tener un modelo con los siguiente campos - por ahora:

* **incidencias**, con los campos descripción, prioridad, urgente y cerrada.
* **citas**, con los camp

Crea el modelo anterior y establece los valores apropiados en el \_\_manifest\_\_.py. Trata de establecer todas las propiedades posibles para estos campos (required, string, help...)

Fichero \_\_manifest\_\_.py

# -\*- coding: utf-8 -\*-

{

    'name': "soporte",

    'summary': """Módulo para la gestion de incidencias de SGE""",

    'description': """

        Módulo para la gestion de incidencias de SGE

    """,

    'author': "Carlos Izquierdo",

    'website': "https://www.carlosig.es",

    'category': 'tools',

'data': [

       'security/ir.model.access.csv',

Fichero models/models.py

**Importación y herencia de clases en Odoo**

1. **Clases principales**:
   * Se importan clases como models y fields, esenciales para definir y extender modelos.
   * Todos los modelos creados deben heredar de models.Model, lo que proporciona las funcionalidades nativas de Odoo.
2. **Variables internas**:
   * Odoo utiliza variables privadas (prefijo \_) que gestionan internamente diversas funcionalidades. Aunque se pueden definir variables personalizadas, es recomendable usar los tipos de campos proporcionados por Odoo para aprovechar al máximo su framework.

**Definición de un modelo**

1. **Creación de un modelo básico**:
   * Se define una clase que extiende de models.Model. Por ejemplo, un modelo llamado soporte.incidencia, donde soporte es el módulo y incidencia es el nombre del modelo.
   * Odoo transforma automáticamente este modelo en una tabla de base de datos en PostgreSQL, con columnas correspondientes a los campos definidos en el modelo.
2. **Nomenclatura del modelo**:
   * El nombre completo del modelo (soporte.incidencia 🡪 modulo.modelo) es clave para referirse a él en todo el desarrollo.

**Definición de campos en un modelo**

1. **Tipos de campos básicos**:
   * Se utilizan los tipos de campos proporcionados por fields en lugar de variables genéricas (por ejemplo, fields.Char para cadenas de texto o fields.Integer para enteros).
   * Ejemplo de campos definidos:
     + **Descripción**: Un campo tipo Char que representa texto breve.
     + **Prioridad**: Un campo tipo Integer que indica niveles de prioridad.
     + **Urgente**: Un campo tipo Selection que permite elegir entre valores predefinidos (Sí o No).
     + **Cerrada**: Un campo tipo Boolean que indica si una incidencia está cerrada.
2. **Configuración de atributos adicionales**:
   * Los campos pueden incluir atributos como valores por defecto, ayudas contextuales (help), longitudes, entre otros.
   * Por ejemplo, para un campo de selección (fields.Selection), se definen pares de valores como tuplas: (valor\_interno, "Valor para mostrar").

**Integración con vistas y ORM de Odoo**

1. **Uso en vistas**:
   * Los campos definidos en un modelo se reflejan automáticamente en las vistas de Odoo, con nombres adaptados a formato capitalizado.
   * Si no se define un atributo como string, el ORM de Odoo utiliza el nombre del campo como etiqueta predeterminada.
2. **Extensibilidad y personalización**:
   * A medida que se desarrolla el modelo, se pueden añadir atributos y configuraciones avanzadas para cada campo.
   * La información adicional sobre estos atributos se encuentra en la documentación oficial de Odoo.

# -\*- coding: utf-8 -\*-

from odoo import models, fields, api

class incidencia(models.Model):

    \_name = 'soporte.incidencia'

    \_description = 'Modelo para la gestion de incidencias'

    descripcion = fields.Char(

        string='Descripcion',

    )

    prioridad = fields.Integer(

        string='Prioridad',

    )

    urgente = fields.Selection(

        string='Urgente',

        selection = [('0','No'), ('1','Si')]

    )

    cerrada = fields.Boolean(

        string='Cerrada',

    )

Fichero views/views.xml

**1. Estructura básica de un módulo**

Antes de comenzar, se recuerda la estructura general aprendida en lecciones anteriores:

* **Elementos de menú:** Opciones que aparecen en la interfaz de usuario.
* **Acciones de ventana:** Conexiones entre los menús y las vistas.
* **Vistas:** Representaciones visuales de los datos de un modelo.

En este caso, se seguirán estos pasos en orden inverso, comenzando por las vistas, continuando con las acciones de ventana y finalizando con los menús.

**2. Creación de vistas**

Se define la vista en el archivo XML con los siguientes elementos clave:

* **Tipo de vista:** Puede ser de tipo lista (tree), formulario (form), entre otros.
* **Nombre:** Debe describir el propósito de la vista, como "Lista de incidencias".
* **Modelo asociado:** En este caso, soporte.incidencia.
* **Identificador:** Es fundamental asignar un identificador único siguiendo las buenas prácticas, como modulo.modelo.nombre.

Por ejemplo, para una vista de lista:

    <record model="ir.ui.view" id="inicidencia\_list">

      <field name="name">Lista de incidencias</field>

      <field name="model">soporte.incidencia</field>

      <field name="arch" type="xml">

        <list>

          <field name="descripcion"/>

          <field name="prioridad"/>

          <field name="urgente"/>

          <field name="cerrada"/>

        </list>

      </field>

    </record>

Las vistas de formulario se definen de forma similar, pero con el tipo form.

**3. Creación de acciones de ventana**

Las acciones de ventana conectan las vistas con los elementos de menú. Para definir una acción de ventana:

* Se utiliza un identificador único.
* Se especifica el modelo asociado y las vistas disponibles, como lista y formulario.

Ejemplo de acción de ventana:

    <record model="ir.actions.act\_window" id="incidencia\_action\_window">

      <field name="name">Lista de incidencia</field>

      <field name="res\_model">soporte.incidencia</field>

      <field name="view\_mode">list,form</field>

    </record>

**4. Creación de elementos de menú**

Los menús permiten acceder a las acciones de ventana desde la interfaz. Se definen en un orden jerárquico:

* **Menú principal:** Visible en la barra superior.
* **Submenús:** Categorías dentro del menú principal.

Ejemplo de menú principal y submenú:

    <!-- Top menu item -->

    <menuitem name="Soporte SGE" id="soporte.menu\_root"/>

    <!-- menu categories -->

    <menuitem name="Incidencias" id="soporte\_incidencias" parent="soporte.menu\_root"/>

    <!-- actions -->

    <menuitem name="Mostrar lista" id="soporte\_incidencias\_list" parent="soporte\_incidencias"

              action="incidencia\_action\_window"/>

fichero security/ir.model.access.csv

En esta parte del documento, se aborda el componente de **Security** en Odoo, específicamente el manejo de los permisos de acceso mediante un archivo CSV. Este archivo, llamado ir.model.access.csv, es un fichero de texto plano separado por comas que contiene la configuración de permisos para los modelos de datos de Odoo.

Para facilitar la edición de estos ficheros, se recomienda instalar extensiones especializadas que ayuden a trabajar con este formato. Esto resulta útil, ya que estos archivos no suelen estar tabulados, lo que puede dificultar la comprensión y edición del contenido. Sin embargo, es importante entender que el archivo define principalmente:

1. **Identificador (ID)**: Un identificador único para el permiso.
2. **Nombre**: Un nombre descriptivo para el acceso.
3. **Modelo**: El modelo de datos al que se aplica el permiso.
4. **Grupo de trabajo**: El grupo al que se asignan los permisos.
5. **Permisos**: Acciones que se permiten realizar, como leer, escribir, crear y borrar.

Por ejemplo, al configurar un permiso en este archivo, se podrían establecer permisos de acceso para un modelo llamado "Soporte de Incidencias". Esto implicaría definir un identificador único, como acceso\_soporte\_incidencias, y asociarlo al modelo correspondiente, utilizando un formato como model\_nombre\_modelo. En este caso, se podría especificar algo como model\_soporte\_incidencias.

En cuanto a los permisos, estos se representan con una serie de valores binarios (1 o 0) que indican si un usuario tiene permiso para realizar las siguientes acciones:

* **Leer**: Consultar registros del modelo.
* **Escribir**: Modificar registros existentes.
* **Crear**: Añadir nuevos registros.
* **Eliminar**: Borrar registros existentes.

Por ejemplo, una configuración típica podría dar todos los permisos (1, 1, 1, 1) a un grupo específico de usuarios, como los usuarios generales.

**Detalles clave a tener en cuenta:**

* El identificador (ID) debe ser único y significativo.
* El nombre es meramente descriptivo y puede ser libremente asignado.
* Los modelos deben ser precedidos por el prefijo model\_ seguido del nombre del modelo en minúsculas y separado por guiones bajos.
* Es fundamental prestar atención a los detalles al definir los permisos, ya que errores pequeños, como un ID mal definido, pueden generar problemas difíciles de localizar.

id,name,model\_id:id,group\_id:id,perm\_read,perm\_write,perm\_create,perm\_unlink

soporte\_incidencia\_acl,soporte\_incidencia,model\_soporte\_incidencia,base.group\_user,1,1,1,1

Tarea entregable

Ahora vosotros deberéis añadir o modificar los arvivos necesarios para añadir los siguientes elementos:

* Creamos un campo titulo de tipo char requerido.
* Modificar el camo descripción de tipo Html también requerido y con un pop-up (help) en el que pongamos el mensaje ‘Explicación de la incidencia ocurrida brevemente
* Modificar el campo prioridad para establecer un valor por defecto de 0 y un mensaje help ‘Establece un valor mayor o igual a 10 para que se considere urgente’
* Modificar el campo urgente a tipo boolean en modo solo lectura “readonly”
* Creamos un campo Binary para archivos adjuntos
* Creamos un campo imagen de tipo Image con ancho y largo máximo 250
* Creamos un campo fecha creacion de tipo Datetime, con valor por defecto la actual “fields.Datetime.now”
* Campo fecha modificacion de tipo Date, con valor por defecto la actual “fields.Date.context\_today”
* Campo ubicación de tipo Selection con una ubicación para cada valor